2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

A Multi-Populated Differential Evolution Algorithm for Solving Constrained Optimization Problem

M. Fatih Tasgetiren, and P. N Suganthan

Abstract-- **This paper presents a multi-populated differential evolution algorithm to solve real-parameter constrained optimization problems. The notion of the "near feasibility threshold" is employed in the proposed algorithm to penalize the infeasible solutions. The algorithm was tested using benchmark instances in Congress on Evolutionary Computation 2006. For these benchmark problems, the problem definition file, codes and evaluation criteria are available in *http://www.ntu.edu.sg/home/EPNSugan*. The performance of the multi-populated differential evolution algorithm is evaluated with the best known or optimal solutions provided in the literature. The experimental results with detailed statistics required for this session show that the proposed multi-populated differential algorithm was able to solve 22 out of 24 benchmark instances to either optimality or best known solutions in the literature. In addition, 6 out of 24 best known solutions are ultimately improved by the proposed multi-populated differential evolution algorithm.**

I. INTRODUCTION

The general constrained optimization problem (P) is concerned with finding $\vec{x}$ so as to:

$$min\, f(\vec{x}) \quad \vec{x} = (x_1, x_2, .., x_n) \in \Re^n$$

where $x \in F \subseteq S$. The objective function $f(\vec{x})$ is defined on the *search space* $S \subseteq \Re^n$ and the set $F \subseteq S$ defines the *feasible region*. In general, S is defined as an n-dimensional space in $\Re^n$ where the domains of the variables are defined by their lower and upper bounds as:

$$l(i) \le x_i \le u(i), \qquad 1 \le i \le n$$

whereas the feasible region F is defined by a set of m additional constraints $(m \ge 0)$ as:

$$g_i(\vec{x}) \le 0 \text{, for } i = 1,..,q \text{ and}$$

$$h_j(\vec{x}) = 0 \text{, for } j = q+1,..,m .$$

A point $\vec{x} \subseteq F$ is called a feasible solution if $\vec{x}$ satisfies both equality and inequality constraints, otherwise $\vec{x}$ is an infeasible solution. The problem P is said to be feasible if there exists at least one feasible point, infeasible otherwise. If $\vec{x}$ is feasible and $g_i(\vec{x}) = 0$, the constraint $g_i(\vec{x}) \le 0$ is said to be *active* at $\vec{x}$. Equality constraints are also *active* at all feasible points.

M. F. Tasgetiren is with the Department of Management, Fatih University, 34500, Buyukcekmece, Istanbul, Turkey (corresponding author), email: ftasgetiren@fatih.edu.tr.

P. N Suganthan is with the School of Electrical and Electronic Engineering Nanyang Technological University, Singapore 639798, email: epsugan@ntu.edu.sg .

Differential evolution (DE) is one of the latest evolutionary optimization methods proposed by Storn and Price [2]. Like other evolutionary-type algorithms, DE is a population-based, stochastic global optimizer.

In a DE algorithm, candidate solutions are represented as chromosomes based on floating-point numbers. In the mutation process of a DE algorithm, the weighted difference between two randomly selected population members is added to a third member to generate a mutated solution followed by a crossover operator to combine the mutated solution with the target solution so as to generate a trial solution. Then a selection operator is applied to compare the fitness function value of both competing solutions, namely, target and trial solutions to determine who can survive for the next generation.

Since DE was first introduced to solve the Chebychev polynomial fitting problem by Storn and Price [2,3], it has been successfully applied in a variety of applications that can be found in Corne et al. [4], Lampinen [5], Babu and Onwubolu [6], and Price et al. [7]. The applications of DE to the constrained optimization problems can also be found in [8, 9, 10, 11, 12, 13, 14, 15, and 16]. The state of the art survey on constrained optimization is given in [17] too.

In this paper we propose a multi-populated differential evolution (MDE) algorithm to solve the benchmark instances designed for this session. Since the benchmark problems are constrained optimization problems, The MDE algorithm employed the notion of the "near feasibility threshold" as a penalty function. The MDE algorithm is not only a parallel implementation of DE algorithm but also dynamic in terms of regrouping the individuals in certain periods of a run. In addition, the individual selection in the mutation phase is from all sub-populations to provide full information exchange among sub-populations at each generation.

The advantages of MDE such as simple concept, immediately accessible for practical applications, simple structure, ease of use, speed to get the solutions, and robustness present itself a good candidate to solve real-parameter constrained optimization problems. Therefore, this paper aims at employing MDE to optimize the benchmark suite presented in the special session of Real Parameter Constrained Optimization in CEC2006.

This paper is organized as follows. Section II gives the methodology of the proposed MDE algorithm, Section III gives the constraint handling method, computational results of test problems are shown in Section IV. Finally, Section V summarizes the concluding remarks.

II. MDE ALGORITHM

In the MDE algorithm developed, the population is divided into smaller sub-populations. Each sub-population conducts its search in parallel, however, in order to avoid the premature convergenge, a regrouping schedule denoted by *R* is introduced in certain periods. Every *R* generations, sub-populations are regrouped to provide information exchange amongst the sub-populations. The individual selection in the mutation phase is also different such that they are selected among sub-populations to provide fully information exchange at each generation.

In addition, a random mutation rate between 0.2 and 0.9 are selected at each generation to generate mutated individuals for sub-populations so that the weighted difference is low for some generations whereas it is high for some other generations.

Currently, there exist several mutation variations in DE. We follow the *DE/rand/1/bin* and *DE/best/1/bin* schemes of Storn and Price [2]. The MDE is presented below.

The MDE algorithm starts with initializing the initial population into a number of sub-populations denoted by *np1*. Each sub-population size is fixed to *np2* individuals. Each individual has an *n*-dimentional vector with parameter values determined randomly and uniformly between predefined search ranges denoted by x_{min} and x_{max} respectively:

$$x_{ijk}^{t} = x_{min} + \left(x_{max} - x_{min}\right) * r_1 \qquad (1)$$

where x_{ijk}^{t} is the j^{th} target individual of i^{th} sub-population with respect to k^{th} dimension at generation t; r_1 is a uniform random number between 0 and 1.

To generate a mutated individual, the MDE mutates vectors from the target sub-populations by adding the weighted difference between two randomly selected target sub-population members to a third member or to a best member in the sub-population. In the MDE algorithm, the following two DE operators are employed to update the dimension values k's of j^{th} individual in the i^{th} sub-population with a flip probability of 0.5 at each generation:

$$\text{Either} \quad v_{ijk}^{t+1} = x_{ajk}^{t} + F\left(x_{bjk}^{t} - x_{cjk}^{t}\right)$$

$$\text{Or} \quad v_{ijk}^{t+1} = g_{ak}^{t} + F\left(x_{bjk}^{t} - x_{cjk}^{t}\right) \qquad (2)$$

where g_{ik}^{t} is the best individual so far in the i^{th} sub-population and a, b, and c are three randomly choosen indexes of the sub-populations such that $\left(a \neq b \neq c \in \left(1,..,np1\right)\right)$, $i = 1,..,np1$, $j = 1,..,np2$, and $k = 1,..,n$. In other words, when mutating the i^{th} sub-population, the individuals are not selected among the members in the i^{th} sub-population because the sub-population size, *np2* is much smaller than the number of sub-populations, *np1* in order to achieve a diversified population at each generation. Instead they are selected from different sub-populations to mutate the individuals for each sub-population. $F > 0$ is a mutation scale factor which affects the differential variation between two individuals. By selecting the individuals from the sub-populations, the information exchange between the sub-populations are automatically achieved at each generation.

Following the mutation phase, the crossover operator is applied to obtain the trial individual such that:

$$u_{ijk}^{t+1} = \begin{cases} v_{ijk}^{t+1}, if & r_{ijk}^{t+1} \leq CR \quad or \quad k = D_k \\ x_{ijk}^{t}, & Otherwise \end{cases} \qquad (3)$$

where the index D_k refers to a randomly chosen dimension ($k=1,..,n$), which is used to ensure that at least one parameter of each trial individual u_{ijk}^{t+1} differs from its counterpart in the previous generation u_{ijk}^{t}. CR is a user-defined crossover constant in the range [0, 1], and r_{ijk}^{t+1} is a uniform random number between 0 and 1. In other words, the trial individual is made up with some parameters of mutant individual, or at least one of the parameters randomly selected, and some other parameters of the target individual.

To decide whether or not the trial individual u_{ijk}^{t+1} should be a member of the target sub-population for the next generation, it is compared to its counterpart target individual x_{ijk}^{t} at the previous generation. The selection is based on the survival of the fitest among the trial sub-population and target sub-population such that:

$$x_{ijk}^{t+1} = \begin{cases} u_{ijk}^{t+1}, if & f\left(u_{ijk}^{t+1}\right) \leq f\left(x_{ijk}^{t}\right) \\ x_{ijk}^{t}, otherwise \end{cases} \qquad (4)$$

During the reproduction of the MDE algorithm, it is possible to extend the search outside of the initial range of the search space. For this reason, parameter values violating the search range are restricted to:

$$x_{ijk}^{t} = q g_{ak}^{t} + \left(1 - q\right) g_{bk}^{t} \qquad (5)$$

where $q=1/2$ in this study and two best solutions from the sub-populations are randomly selected ($a \neq b \in \left(1,..,np1\right)$) and averaged to keep useful information about the best solutions obtained so far in the whole population.

Regrouping schedule is also important in the design of the MDE algorithm. In order to take advantages of the best information obtained during the search process, the i^{th} sub-population is regrouped in every $R=100$ generations in such a way that the j^{th} individual is regrouped by averaging the dimension values of two randomly selected best individuals of the sub-populations as follows:

$$x_{ijk}^{t} = qg_{ak}^{t} + (1-q)g_{bk}^{t} \qquad (6)$$

where $q=1/2$ in this study and g_{ak}^{t} and g_{bk}^{t} are the best individuals for a^{th} and b^{th} sub-populations such that $a \neq b \in (1,..,np1)$ and $i = 1,..,np1$, $j = 1,..,np2$, $k = 1,..,n$. It should be noted that some other crossover operators could be used instead of the averaging operator which pushes the population to the center of the search range. The pseudo code of the MDE algorithm is given in Fig. 1.

```
Initialize parameters
Initialize target sub-populations
Evaluate target sub-populations
Do {
    Obtain mutant sub-populations
    Obtain trial sub-populations
    Evaluate trial sub- populations
    Make selection
  If (mod R) Regroup sub-populations
While (Not Termination)
```

Fig. 1. MDE Algorithm.

III. CONSTRAINT HANDLING

Since evolutionary operators may generate infeasible solutions, care must be taken with them violating the constraints. There exists different approaches to handle the constraints [17]. One popular way is to penalize the infeasible solutions based on their distance from feasibility. The adaptive penalty approach presented in [1] is used in this paper to handle the constraints. In [1], the adaptive penalty function introduces the notion of a "near feasibility threshold" (*NFT*) corresponding to a "promising region" beyond the feasible region. The *NFT* is defined as a threshold distance from feasible region such that the search within feasible region and the *NFT*-neighborhood of the feasible region is encouraged whereas it is discouraged beyond that threshold. Additionally, an adaptive term is added to the penalty function to consider the gap between the best feasible value and best infeasible value found so far. Then the adaptive penalty function is defined as:

$$f_p(x) = f(x) + (f_{feas} - f_{all}) \sum_{i=1}^{n} \left(\frac{p_i(x)}{NFT_i} \right)^{\alpha_i} \qquad (7)$$

where f_{all} denotes the unpenalized value of the best solution found yet and f_{feas} denotes the value of the best feasible solution yet found. As noted in [17], the adaptive term may lead to zero-or over-penalty. For this reason, only the dynamic part of the above penalty function with *NFT* threshold is used in this paper as follows:

$$f_p(x) = f(x) + \sum_{i=1}^{q} \left(\frac{G_i(x)}{NFT_i} \right)^{\alpha} + \sum_{j=q+1}^{m} \left(\frac{H_j(x)}{NFT_j} \right)^{\alpha} \qquad (8)$$

where $G_i(x) = \begin{cases} g_i(x) & if \quad g_i(x) > 0 \\ 0 & if \quad g_i(x) \leq 0 \end{cases}$ and

$$H_j(x) = \begin{cases} fabs(h_j(x)) & if \quad fabs(h_j(x)) - \varepsilon > 0 \\ 0 & if \quad fabs(h_j(x)) - \varepsilon \leq 0 \end{cases}$$

The general form of the *NFT* is given by: $NFT = \frac{NFT_0}{1 + \lambda * t}$ where NFT_0 is an upper bound for the NFT ; lamda is a user-defined positive parameter; and t is the generation counter.

IV. COMPUTATIONAL RESULTS

The MDE algorithm was coded in C and run on an Intel P4 1.33 GHz Laptop PC with 256MB memory. Regarding the MDE parameters, the mutation scale factor (*F*) is randomly and uniformly determined between 0.2 and 0.9 at each generation such that $F \in (0.2,0.9)$. The crossover rate (*CR*) is taken as 0.9. The number of sub-populations *np1* and the sub-population size *np2* are taken as 20 and 5, respectively. Every *R*=100 generations, the sub-populations are regrouped. ε is predefined as 0.0001 for this session.

The severity parameter, α , and the positive constant, λ , of the penalty function are taken as 2 and 0.04, respectively. The choice of initial value of NFT_0 threshold is the key to success of the algorithm. The larger the NFT_0 threshold distance, the higher the probability that the algorithm results in an infeasible solution. Since the equality constraints are converted to the inequality constraints by subtracting ε from the absolute value of the constraint value and ε is predefined, the NFT_0 is choosen as 1e-16, which is much smaller than ε such that the NFT value will always be very small, hence the magnitute of the penalty term will be very high depending on the distance from feasibility. In fact, the NFT_0 threshold for the equality constraints is given by ε implicitly. Since the solutions inside this distance are treated as feasible, the solutions beyond this distance will be so much penalized through the use of a very small NFT_0. So it is assured that the search process always tends to favor the feasible solutions while benefiting from the infeasible solutions with a dynamic NFT depending on the generation counter. It is obvious that this simple choice reduces the efforts of tuning the parameter of the penalty function, which is a very desirable property of any stochastic algorithm. In other words, the near feasibility threshold is not only used for penalizing infeasible solutions, but also for preserving the feasibility of the solutions in the population.

The maximum number of function evaluations is fixed at 5e3, 5e4 and 5e5 FES. The MDE algorithm was run on the 24 benchmark functions and the performance evaluation of the MDE algorithm is also conducted through the guidelines described for this session. 25 replications are conducted for each benchmark function to record the error values, $f(x) - f(x^*)$, after 5e3 FES, 5e4 FES, and 5e5 FES. The error values achieved at different FES levels are given in details in Tables I to IV. Table V shows the number of FES to achieve the fixed accuracy level $(f(x) - f(x^*) \leq 0.0001)$, success rate, feasible rate and success performance of the MDE algorithm. Finally, the complexity of the MDE algorithm is given in Table VI.

As shown in Table V, except for g20 and g22, the MDE algorithm was able to find feasible and successful solutions for 22 benchmark functions. Ultimately, the best known or optimal solution of six functions is slightly improved by the proposed MDE algorithm.

The new best known solution for g03 is given as:

	g03	
x	0.3162435743510031427	0.3162435770867201290
	0.3162435778836074074	0.3162435769431206634
	0.3162435800148344350	0.3162435770323555051
	0.3162435758520770746	0.3162435772352355512
	0.3162435766144676696	0.3162435770853171957
h	0.0000999999999999898659	
	0.0000150911731544578486 8	
f	-1.0005001000100031128	

The new best known solution for g07 is given as:

	g07	
x	2.171996373980543904	2.363682966322170564
	8.773925735944821724	5.095984481018770218
	0.9906547697995234936	1.43057399359684756
	1.321644213795827305	9.828725812650494831
	8.280091681362916844	8.37592667380728706
g	0.0000000000000000000000000000000000	
	-0.000000000000055455640080026569200 00	
	-0.0000000000000014224732503009818180 0	
	-0.0000000000000006912873051767576272 0	
	-6.1485033729608211670000000000000000	
	-50.023961841035145420000000000000000 0	
f	24.30620906817981108	

The new best known solution for g10 is given as:

	g10	
x	579.306689824117484	1359.970662220993063
	5109.970668483558256	182.0177000320194622
	295.601173260657788	217.9822999679805093
	286.4165267713616458	395.6011732606577311
g	-0.00000000000000000502527706947031305 2	
	-0.00000000000000000502527706947031305 2	
	-0.00000000000000000547630517322428289 60	
	-0.000000000002458477865729946643000 00	
	-0.000000000000198951966012828052000 00	
	-0.000000000005115907697472721338000 00	
f	7049.248020528668348	

The new best known solution for g13 is given as:

	g13	
x	-1.717142241679951775	1.595721242405279794
	1.827250237561395263	0.7636598739018681803
	0.7636598750090957122	
h	0.0000999999999999994735322	
	0.0000999999999999993629436	
	0.0000999999999999963033251	
f	0.0539415140418979047 2	

The new best known solution for g14 is given as:

	g14	
x	0.0406684017484546245 2	0.1477212596955897461
	0.783205696687242714	0.0014143418830258771 94
	0.4852936335361681674	0.0006931829776835397904
	0.02740520806695417866	0.01795096602430438018
	0.03732681335665093136	0.09688450250819682918
h	0.0000999999999999998768554	
	0.0000999999999999999191393	
	0.0000999999999999989878097	
f	-47.76488845949148754	

The new best known solution for g21 is given as:

	g21	
x	193.7245100700349667	6.898663038625076252e-27
	17.31918872940849141	100.04789780138702326
	6.684451853623778916	5.991684284442648334 0
	6.214516488860703624	
g	-0.00000000000000097144514654701197290 0	
h	0.00009999999999992661514630000000000 0	
	0.00009999999999997669419800000000000 0	
	0.00009999999999997218391700000000000 0	
	0.00009999999999995670151000000000000 0	
	0.00009999999999995917349100000000000 0	
f	193.7245100700349667	

It can be seen from Table V that the MDE algorithm performed very good in terms of the feasible and success rates. Among 24 problems, the feasible rate of 20 functions was 100%; for function g13 and g17, it was 88% and 96%, respectively; however it failed for the g20 and g22. In terms of the success rate, it was 100% for 16 problems; 92%, 84%, 96%, 48%, 28%, and 68% for the functions g02, g03, g11, g13, g17, and g21, respectively. Again, it failed for the functions g20 and g22. It should be noted that that the performance of the MDE algorithm is much better than the very recent "cultural differential evolution algorithm" for the first 13 functions presented in [16] . The overall feasible and success rate were %91 and %84, respectively. It is worth saying that a little effort of tuning parameters of the MDE algorithm would lead to the success rate of 100% for the functions g02, g03, g11, g13, g17, and g21 . Hovewer, we report our results based on the same set of parameters.

V. CONCLUSIONS

In this paper, a multi-populated differential evolution algorithm is presented to solve the benchmark problems designed for the special session on real-parameter constrained optimization problems at CEC2006. The notion of the "*near feasibility threshold*" is employed to penalize the infeasible solutions. To the best of our knowledge, this is the first reported application of NFT applied to the constrained optimization problems.

The MDE algorithm performed well enough for 22 out of 24 benchmark instances. Ultimately, 6 out of 24 benchmark instances were slightly improved by the MDE algorithm.

However, since we do not know how the other sophisticated algorithms performed for the benchmarks presented for this session, only the computational results with the necessary statistics are presented to be compared with the competing DE and non-DE algorithms during the conference and afterwards.

BIBLIOGRAPHY

[1] Smith A. E and Tate D. M. (1993) "Genetic Optimization Using a Penalty Function", *Proc. of the Fifth International Conference on genetic Algorithms*, S. Forrest (Ed), Morgan Kaufmann, pp.499-503.

[2] Storn, R. and Price, K. (1997) "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Space," *Journal of Global Optimization*, vol. 11, pp. 341-359.

[3] Storn, R. and Price, K. (1995) "Differential Evolution – a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces," *Technical Report TR-95-012*, ICSI, 1995.

[4] Corne, D., Dorigo, M., and Glover, F. (eds.) (1999) "Part Two: Differential Evolution," *New Ideas in Optimization*, McGraw-Hill, pp. 77-158.

[5] Lampinen, J. (2001) "A Bibliography of Differential Evolution Algorithm," *Technical Report*, Lappeenranta University of Technology, Department of Information Technology, Laboratory of Information Processing.

[6] Babu, B. V. and Onwubolu, G. C. (eds.) (2004) *New Optimization Techniques in Engineering*, Springer Verlag.

[7] Price, K., Storn, R., and Lampinen, J. (2005) "*Differential Evolution – A Practical Approach to Global Optimization*", Springer-Verlag.

[8] Storn, R. (1999) "System Design by Constraint Adaptation and Differential Evolution," *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 22-34.

[9] Lampinen J. (2002) "A Constraint Handling approach for the Differential evolution Algorithm" *Proc. of the Congress on Evolutionary Computation (CEC2002)*, pp. 1468-1473.

[10] Koziel S. And Michalewicz Z. (1999) "Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization" *Evol. Comput.*, 7(1) pp. 19-44.

[11] Lampinen J. (2001) "Multi-Constrained Optimization by the Differential Evolution," *Proc. of the IASTED International Conference Artificial Intelligence Applications (AIA 2001)*, pp. 177-184.

[12] Lampinen J. (2001) "Solving Problems Subject to Multiple Nonlinear Constraints by the Differential Evolution", *Proc. of the MENDEL2001, 7th International Conference on Soft Computing,* pp. 50-57.

[13] Lin Y-C, Hwang K-S, and Wang F-S, (2002) "Hybrid Differential Evolution with Multiplier updating method for Nonlinear Constrained Optimization" ,*Proc. of the Congress on Evolutionary Computation (CEC2002)*, pp. 872-877.

[14] Chiou J-P and Wang F-S. (1999) "Hybrid Method of Evolutionary Algorithms for Static and Dynamic Optimization Problems with Applications to a Fed-Batch fermantation Process", *Computers and Chemical Engineering*, 23, pp. 1277-1291.

[15] Sarimveis H and Nikolakopoulos A. (2005) "A Line Up Evolutionary Algorithm for Solving Nonlinear Constrained Optimization Problems", *Computers & Operations Research,* 32, pp. 1499-1514.

[16] Becerra R. L and Coello Carlos A. Coello. (2005) "Cultural Differential Evolution for Constrained Optimization", *Comput. Methods Appl. Mech. Engrg.*, (To appear).

[17] Coello Carlos A. Coello. (2002) "Theoretical and Numerical Constraint-Handling Techniques Used with Evolutionary Algorithms: A Survey of the State of the Art", *Comput. Methods Appl. Mech. Engrg.*, 191(11-12), pp. 1245-1287.

PC Configuration:

System:	Microsoft Windows XP Professional Version 2003
Computer:	Intel P4 1.33 GHz Laptop PC with 256MB Memory
Language:	C

TABLE I. ERROR VALUES ACHIEVED AT 5E3 FES, 5E4 FES, 5E5 FES FOR PROBLEMS 1-6

FES		g01	g02	g03	g04	g05	g06
	Best	2.6354E+00	4.7077E-01	5.2551E-01	8.0730E+00	-1.1271E+00	1.5181E+00
	Median	3.7956E+00	5.0779E-01	9.3137E-01	2.4846E+01	1.1835E+02	1.4357E+01
	Worst	5.4540E+00	5.4036E-01	9.9940E-01	6.3048E+01	6.5218E+02	1.0935E+02
5 x e3	c	0,0,0	0,0,0	0,0,1	0,0,0	0,3,3	0,0,0
	v	0.0000E+00	0.0000E+00	1.0663E-04	0.0000E+00	3.9375E-02	0.0000E+00
	Mean	3.7731E+00	5.0642E-01	8.7345E-01	2.6166E+01	1.1711E+02	1.7814E+01
	Std	6.6308E-01	2.0863E-02	1.3743E-01	1.2091E+01	1.4022E+02	2.2248E+01
	Best	3.6835E-06	7.3193E-02	3.4330E-01	3.6380E-12	1.1214E-07	1.0914E-11
	Median	1.8665E-05	1.3019E-01	5.6809E-01	3.6380E-12	6.8566E+01	1.0914E-11
	Worst	6.9116E-05	1.8427E-01	7.8985E-01	3.6380E-12	3.9622E+02	1.0914E-11
5 x e4	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Mean	1.9333E-05	1.2841E-01	5.5140E-01	3.6380E-12	8.6648E+01	1.0914E-11
	Std	1.4811E-05	2.3721E-02	1.2419E-01	0.0000E+00	9.5337E+01	0.0000E+00
	Best	0.0000E+00	2.3414E-06	**-3.1086E-15**	3.6380E-12	0.0000E+00	1.0914E-11
	Median	0.0000E+00	3.5608E-06	**-2.8866E-15**	3.6380E-12	0.0000E+00	1.0914E-11
	Worst	0.0000E+00	1.1014E-02	**4.0082E-01**	3.6380E-12	1.8190E-12	1.0914E-11
5 x e5	c	0,0,0	0,0,0	**0,0,0**	0,0,0	0,0,0	0,0,0
	v	0.0000E+00	0.0000E+00	**0.0000E+00**	0.0000E+00	0.0000E+00	0.0000E+00
	Mean	0.0000E+00	8.0224E-04	**2.5237E-02**	3.6380E-12	7.2760E-14	1.0914E-11
	Std	0.0000E+00	2.7802E-03	**8.3681E-02**	0.0000E+00	3.6380E-13	0.0000E+00

TABLE II. ERROR VALUES ACHIEVED AT 5E3 FES, 5E4 FES, 5E5 FES FOR PROBLEMS 7-12

FES		g07	g08	g09	g10	g11	g12
	Best	1.2968E+01	1.1338E-14	2.3289E+00	1.5761E+03	8.1250E-05	2.2384E-06
	Median	2.5799E+01	1.0672E-13	5.5382E+00	1.5761E+03	3.9238E-02	1.0835E-04
	Worst	4.8760E+01	1.7404E-12	9.2047E+00	3.2505E+03	2.3421E-01	4.0091E-04
5 x e3	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Mean	2.5561E+01	2.4358E-13	0.0000E+00	2.3507E+03	5.2665E-02	1.1608E-04
	Std	8.8296E+00	3.8274E-13	2.1383E+00	4.0937E+02	6.1645E-02	1.1586E-04
	Best	1.6569E-04	4.1633E-17	4.5475E-13	2.3793E-02	0.0000E+00	0.0000E+00
	Median	4.3014E-04	5.5511E-17	4.6612E-12	5.7534E-02	0.0000E+00	0.0000E+00
	Worst	8.0837E-04	5.5511E-17	3.2060E-11	3.4142E-01	7.9512E-03	0.0000E+00
5 x e4	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Mean	4.1842E-04	4.9405E-17	7.5033E-12	8.1478E-02	3.3218E-04	0.0000E+00
	Std	1.7621E-04	7.0308E-18	7.6773E-12	7.3937E-02	1.5889E-03	0.0000E+00
	Best	**-1.8829E-13**	4.1633E-17	1.1369E-13	**-1.8190E-12**	0.0000E+00	0.0000E+00
	Median	**-1.8474E-13**	4.1633E-17	1.1369E-13	**-9.0949E-13**	0.0000E+00	0.0000E+00
	Worst	**-1.7764E-13**	4.1633E-17	1.1369E-13	**-9.0949E-13**	3.5319E-04	0.0000E+00
5 x e5	c	**0,0,0**	0,0,0	0,0,0	**0,0,0**	0,0,0	0,0,0
	v	**0.0000E+00**	0.0000E+00	0.0000E+00	**0.0000E+00**	0.0000E+00	0.0000E+00
	Mean	**-1.8531E-13**	4.1633E-17	1.1369E-13	**-1.0186E-12**	1.4128E-05	0.0000E+00
	Std	**3.0215E-15**	0.0000E+00	0.0000E+00	**3.0165E-13**	7.0638E-05	0.0000E+00

TABLE III. ERROR VALUES ACHIEVED AT 5E3 FES, 5E4 FES, 5E5 FES FOR PROBLEMS 13-18

FES		g13	g14	g15	g16	g17	g18
	Best	3.9089E-01	1.7728E+00	-6.7718E-03	1.3919E-02	1.6035E+01	2.2474E-01
	Median	8.6864E-01	3.9239E+00	2.0164E+00	2.2816E-02	9.4054E+01	4.6358E-01
	Worst	8.5662E+00	6.5406E+00	7.7724E+00	4.4351E-02	3.7153E+02	6.6186E-01
5 x e3	c	0,3,3	0,2,3	0,0,2	0,0,0	0,3,4	0,0,0
	v	2.0099E-02	3.3576E-02	7.7932E-03	0.0000E+00	2.8290E-01	0.0000E+00
	Mean	1.1078E+00	3.9181E+00	2.0012E+00	2.4650E-02	1.3784E+02	4.4001E-01
	Std	1.5716E+00	1.4211E+00	1.8802E+00	8.1356E-03	1.1277E+02	1.0596E-01
	Best	4.6548E-01	-3.9226E-04	0.0000E+00	6.6613E-15	1.8555E+01	4.8803E-06
	Median	8.4619E-01	-3.3501E-04	1.0851E+00	1.0658E-14	1.0572E+02	2.7798E-05
	Worst	9.4526E-01	1.8008E-03	4.9573E+00	4.5075E-14	3.8915E+02	3.7752E-04
5 x e4	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,2	0,0,0
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	4.2848E-04	0.0000E+00
	Mean	7.6797E-01	-2.0875E-04	1.0890E+00	1.2967E-14	1.3254E+02	5.4155E-05
	Std	1.5299E-01	4.3915E-04	1.2278E+00	8.0226E-15	1.0104E+02	8.4229E-05
	Best	**-9.7145E-17**	**-3.9961E-04**	0.0000E+00	5.3291E-15	3.6380E-12	3.3307E-16
	Median	**3.8486E-01**	**-3.9961E-04**	0.0000E+00	5.3291E-15	7.4058E+01	4.4409E-16
	Worst	**9.2964E-01**	**-3.9961E-04**	2.1514E-05	5.3291E-15	7.4058E+01	4.4409E-16
5 x e5	c	**0,0,0**	**0,0,0**	0,0,0	0,0,0	0,0,0	0,0,0
	v	**0.0000E+00**	**0.0000E+00**	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Mean	**2.5364E-01**	**-3.9961E-04**	8.6055E-07	5.3291E-15	5.1353E+01	4.2633E-16
	Std	**2.8719E-01**	**7.9441E-15**	4.3027E-06	0.0000E+00	3.4111E+01	4.1541E-17

TABLE IV. ERROR VALUES ACHIEVED AT 5E3 FES, 5E4 FES, 5E5 FES FOR PROBLEMS 19-24

FES		g19	g20	g21	g22	g23	g24
	Best	9.1894E+01	5.4329E+00	4.5116E+01	2.8012E+03	-2.9197E+02	2.7211E-06
	Median	1.3470E+02	1.0151E+01	5.1294E+02	1.1107E+04	2.0193E+02	2.9456E-05
	Worst	1.8707E+02	1.5479E+01	7.9312E+02	1.6276E+04	7.5439E+02	1.1482E-04
5 x e3	c	0,0,0	2,19,20	0,5,5	19,19,19	0,5,5	0,0,0
	v	0.0000E+00	6.7414E+00	9.4217E-02	1.2721E+06	9.3925E-02	0.0000E+00
	Mean	1.3450E+02	1.0052E+01	4.5213E+02	1.0303E+04	1.8403E+02	3.3281E-05
	Std	2.6993E+01	2.2768E+00	2.3448E+02	3.5482E+03	2.5804E+02	2.8933E-05
	Best	4.6286E-02	6.7495E-02	6.4811E-06	2.7125E+03	7.8398E+00	7.1054E-14
	Median	1.1539E-01	1.0151E+01	1.1983E+02	9.6011E+03	4.1024E+01	7.1054E-14
	Worst	2.3681E-01	1.5479E+01	1.3098E+02	1.8314E+04	1.2776E+02	7.1054E-14
5 x e4	c	0,0,0	2,19,20	0,0,0	18,19,19	0,0,0	0,0,0
	v	0.0000E+00	6.7414E+00	0.0000E+00	8.6352E+01	0.0000E+00	0.0000E+00
	Mean	1.2716E-01	9.8373E+00	7.9457E+01	9.6146E+03	4.6519E+01	7.1054E-14
	Std	5.3918E-02	2.8984E+00	5.9352E+01	5.2790E+03	3.1193E+01	0.0000E+00
	Best	2.8422E-14	6.7495E-02	**-2.8422E-14**	1.1823E+03	0.0000E+00	7.1054E-14
	Median	3.5527E-14	1.0151E+01	**1.4211E-13**	8.7919E+03	0.0000E+00	7.1054E-14
	Worst	2.2737E-13	1.5479E+01	**1.3098E+02**	1.6821E+04	2.2737E-13	7.1054E-14
5 x e5	c	0,0,0	2,19,20	**0,0,0**	16,19,19	0,0,0	0,0,0
	v	0.0000E+00	6.7414E+00	**0.0000E+00**	3.9475E+00	0.0000E+00	0.0000E+00
	Mean	4.6612E-14	9.8373E+00	**4.1913E+01**	8.6564E+03	1.8190E-14	7.1054E-14
	Std	3.9831E-14	2.8984E+00	**6.2358E+01**	4.8022E+03	5.1159E-14	0.0000E+00

TABLE V. NUMBER OF FES TO ACHİEVE FIXED ACCURACY LEVEL

Prob.	Best	Median	Worst	Mean	Std	FR	SR	SP
g01	37633	43794	49654	43430	3015.29	100%	100%	43430.00
g02	260388	280272	301048	280573	12324.00	100%	92%	304970.65
g03	119629	210525	277057	209298	46084.00	100%	84%	24860.71
g04	19717	20823	22609	20883	794.00	100%	100%	20883.00
g05	39701	236872	477209	216469	126986.00	100%	100%	216469.00
g06	9872	10550	11528	10574	430.00	100%	100%	10574.00
g07	52438	57079	63445	57400	2370.00	100%	100%	57400.00
g08	990	1632	2068	1515	336.00	100%	100%	1514.50
g09	18608	20814	24025	21044	1123.00	100%	100%	21044.00
g10	42610	48508	60924	48628	4315.00	100%	100%	48628.00
g11	3884	21027	55738	22422	12610.00	100%	96%	23356.25
g12	1044	4227	6510	4238	1517.00	100%	100%	4238.00
g13	268723	351183	429252	356433	44206.00	88%	48%	742568.75
g14	34099	42462	58180	42715	5424.00	100%	100%	42715.00
g15	12240	231550	490328	200174	125815.00	100%	100%	200174.00
g16	11036	13135	14418	13063	724.00	100%	100%	13063.00
g17	117917	206674	270372	204791	47482.00	96%	28%	731396.43
g18	36211	42550	57603	44045	5141.00	100%	100%	44045.00
g19	102385	115054	146392	118274	11533.00	100%	100%	118274.00
g20	*	*	*	*	*	*	*	*
g21	42542	88066	303029	142159	102300.00	100%	68%	209057.35
g22	*	*	*	*	*	*	*	*
g23	109493	210661	293966	210661	51262.00	100%	100%	210661.00
g24	3623	4371	5099	4342	444.60	100%	100%	4342.20

TABLE VI. COMPLEXITY OF THE DE ALGORITHM

T1(s)	T2(s)	Complexity
0.0643	0.1568	1.4385